

dbms overview

Top-level Goals of DBMSs

Provide solutions to data processing problems that applications developers would otherwise have to solve by themselves:

- **Provide meaning-based view of data**
Shield from irrelevant detail (i.e., create an abstract view)
- **Support operations on data**
Queries & updates
- **Provide data control**
Integrity, protection, concurrency & recovery

Purpose of DBMSs

Provide solutions to data processing problems that applications developers would otherwise have to solve by themselves:

- **Data redundancy & inconsistency**
Multiple file formats, duplication of information in different files
- **Difficulty in accessing data**
Need to write a new program to carry out each new task
- **Integrity problems**
 - Integrity constraints (e.g. $\text{account balance} > 0$) become “buried” in program code rather than being stated explicitly
 - Hard to add new constraints or change existing ones

Purpose of DBMSs

Provide solutions to data processing problems that applications developers would otherwise have to solve by themselves:

- Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
 - Concurrent access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people read & update balance at the same time

Purpose of DBMSs

Provide solutions to data processing problems that applications developers would otherwise have to solve by themselves

- **Security problems**

Provide user access to some, but not all, data

core database issues

Data models, query languages
Database design
Query processing
Storage management
Transaction management
Concurrency control

Data Models

- A way to *logically* represent data
The physical implementation (i.e., how the data are actually stored) is hidden
- Examples of data models
 - Relational
 - Entity-Relationship (E-R)
 - Object-relational
 - XML (Extensible Markup Language)

Data Models

Relational Model

- Represent data as tables

Customer table

CustID	Name	Street	City
1	Fred Flintstone	First Av	SD
2	Barney Rubble	Main Street	SD
3	Maggie Simpson	Cartoon Way	SF
4	James Bond	Dangerous Av	NY

Account table

Account Num	Balance
A1	500
A2	700
A3	1000

Depositor table

CustID	AccountNum
1	A1
1	A2
4	A3

Relational Model: SQL

- Query language for relational databases
- Is declarative (non-procedural)
You specify the desired result but not how to compute it

Example: Find the name of the customer with customer-id 1

```
select  customer.name  
from    customer  
where   customer.cust_id = '1'
```

Relational Model: SQL

- Query language for relational databases
- Is declarative (non-procedural)
You specify the desired result but not how to compute it

Example: Find the balances of all accounts held by the customer with customer-id 1

```
select  account.balance
from    depositor, account
where   depositor.custid = '1' and
          depositor.accountnum = account.accountnum
```

Relational Model: SQL

Application programs generally access databases through:

- Language extensions
to allow embedded SQL
- Application program interface (e.g., ODBC/JDBC)
which allow SQL queries to be sent to a database

Data Models

Entity Relationship (E-R) Model

Models an application as a collection of:

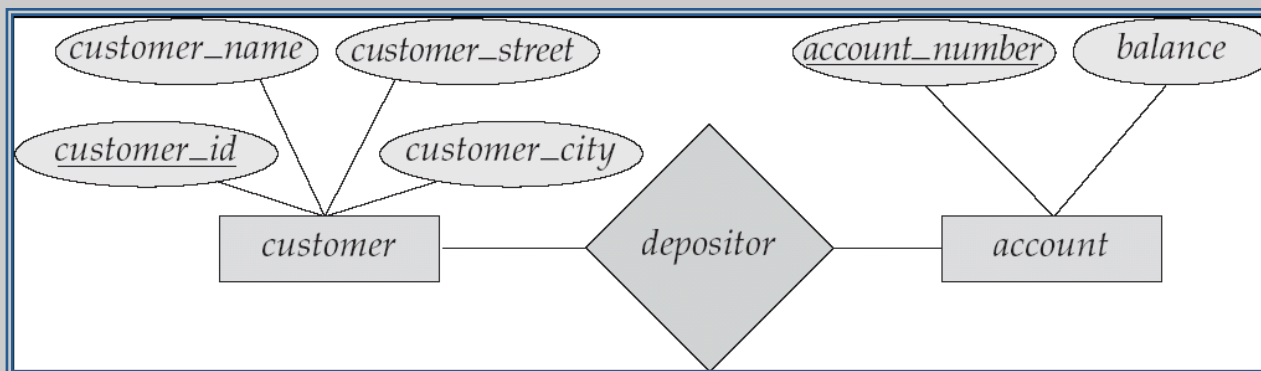
- Entities

“Things”/“objects” in the enterprise that is distinguishable from other objects (and is described by a set of attributes)

- Relationships

Associations among several entities

Represented as an E-R diagram:



Object-Relational Model

- **Extend the relational data model**
include object orientation and constructs to deal with added data types
- **Allow attributes of tuples to have complex types**
including non-atomic values such as nested relations
- **Preserve relational foundations**
in particular the declarative access to data, while extending modeling power
- **Provide backwards compatibility**
with existing relational languages

Data Models

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- Great way to exchange data (not only docs)
based on the ability to specify new tags, and to create nested tag structures
- Basis for a new generation of data exchange formats
- Wide variety of XML tools available
for parsing, browsing and querying XML documents/data

Database Design

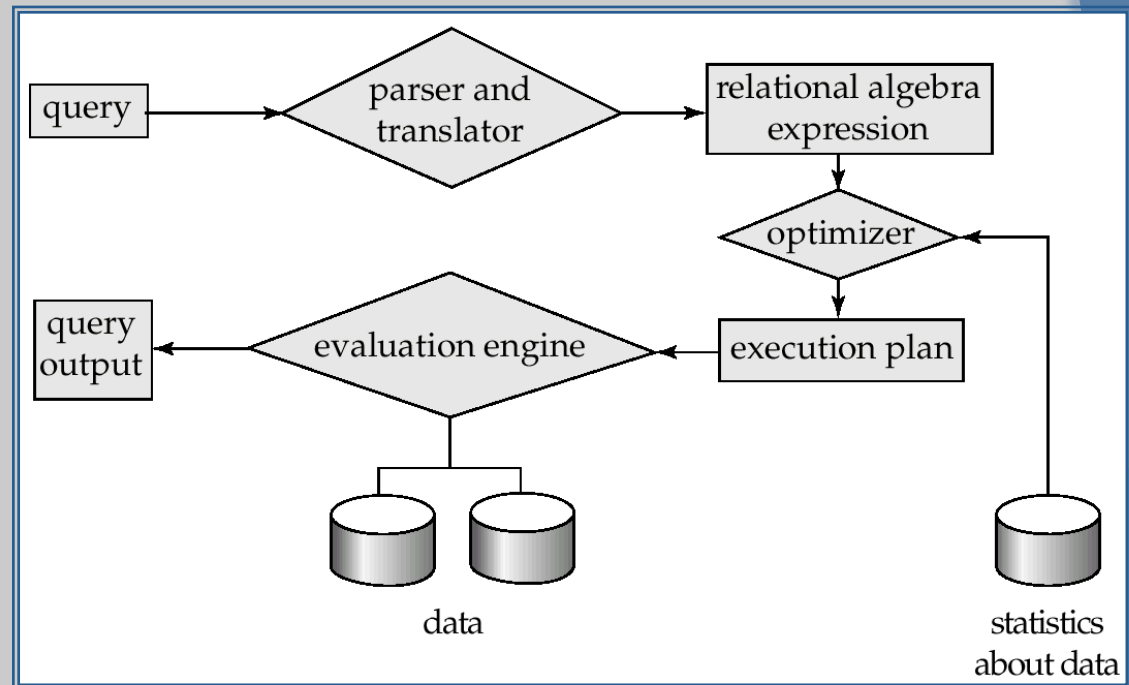
Designing the general structure of the database:

- **Conceptual Design**
Captures data requirements
(i.e., which information should be present)
e.g., for relational model, the conceptual design can be done through the E-R Model
- **Logical Design**
Translate the conceptual design to the chosen data model
e.g., define the schema of the relational tables
- **Physical Design**
Decide on the physical layout of the database
e.g., define indices

Query Processing

3 steps:

- Parsing & Translation
- Optimization
- Evaluation



Query Processing

- Alternative ways of evaluating a given query
 - Equivalent formulations
 - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
 - Depends critically on statistical information about relations which the database must maintain
 - Need to estimate statistics for intermediate results to compute cost of complex expressions

Storage Management

- **Storage manager**
is a module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the following tasks:
Interaction with the file manager
Efficient storing, retrieving and updating of data
- **Issues:**
File organization
Indexing

Transaction Management

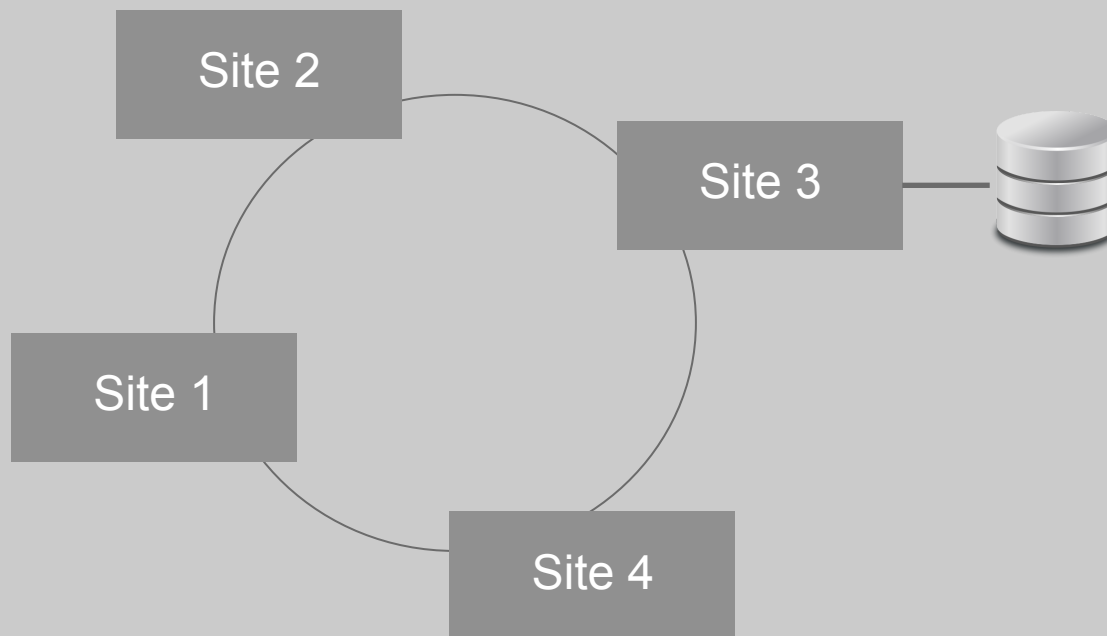
- **Transaction**
is a collection of operations that performs a single logical function in a database application
- **Transaction management**
ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency control**
manages the interaction among the concurrent transactions, to ensure the consistency of the database

Database Architecture

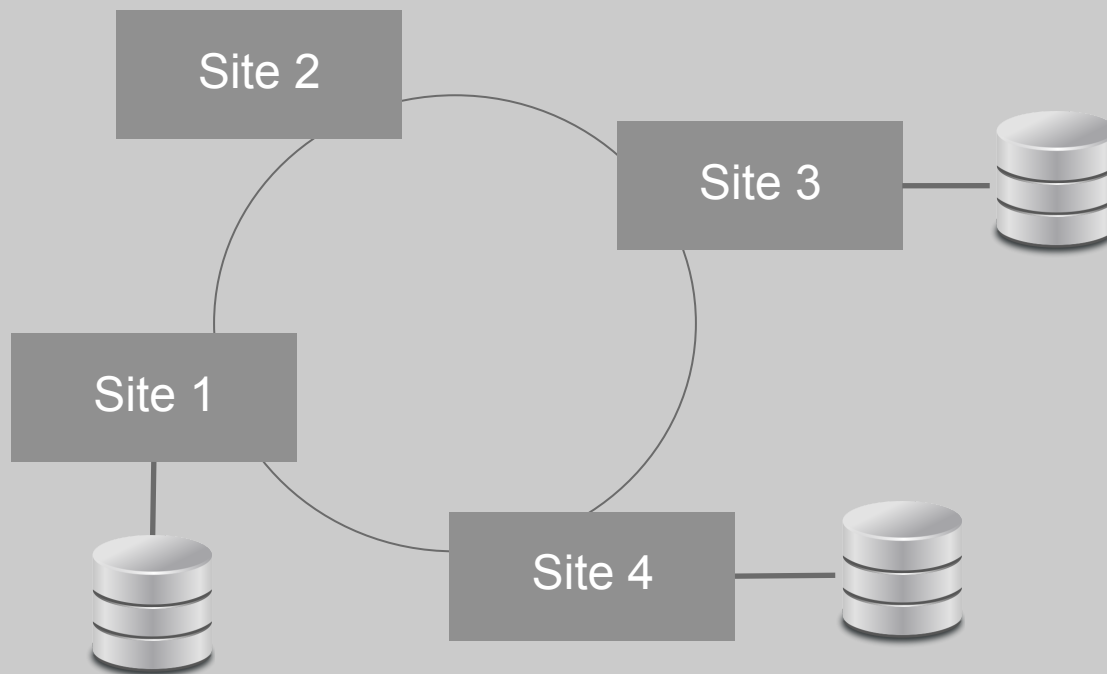
The architecture of a database systems is influenced by the underlying computer system on which the database is running:

- Centralized vs Distributed
- Parallel (multi-processor)
e.g., Map-Reduce

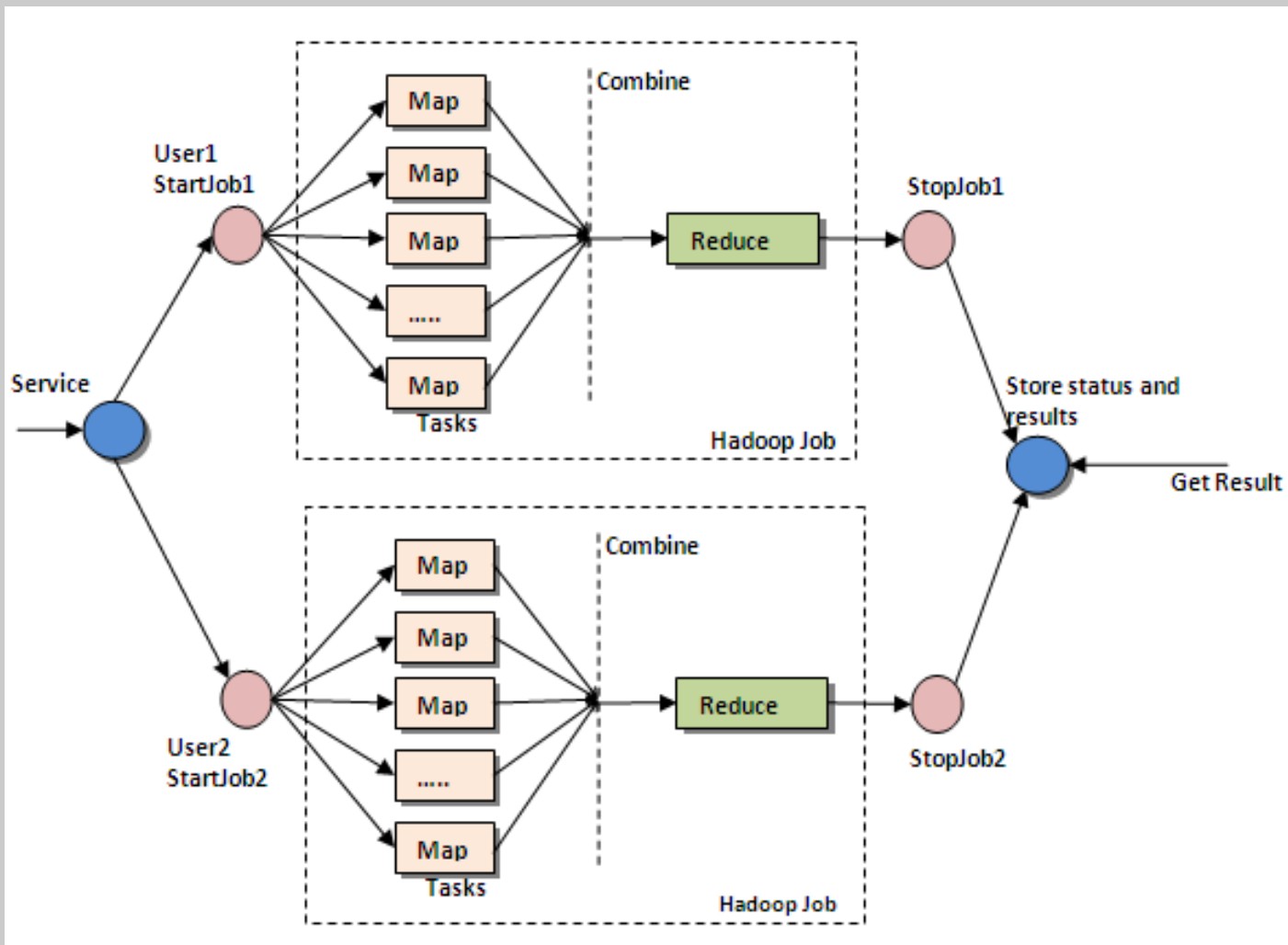
Centralized DBMS



Distributed DBMS



Map-Reduce



Purpose of DBMSs revisited

Which feature of a DBMS tackles each of the following problems?

- Data redundancy & inconsistency ?
- Difficulty in accessing data ?
- Integrity problems ?
- Atomicity of updates ?
- Concurrent access by multiple users ?
- Security problems ?